**Amendments to the Specification:**

Rewrite the title at page 1, lines 1 to 2 as follows:

A1

MICROPROCESSOR WITH AN INSTRUCTION <u>SEQUENTIALLY ADJACENT TO A BRANCH INSTRUCTION</u> FOR ADDING A CONSTANT TO A PROGRAM COUNTER

Rewrite the paragraph at page 1, lines 13 to 17 as follows:

A2

This application claims priority under 35 USC §119(e)(1) of Provisional Application No. 60/183,527, filed February 18, 2000 ~~(TI-30302PS)~~ of Provisional Application No. 60/173,795, filed December 30, 1999 ~~(TI-26738P1)~~ and of Provisional Application No. 60/183,357, filed February 18, 2000 ~~(TI-26738P2).~~

Rewrite the paragraph at page 6, lines 8 to 9 as follows:

A3

~~Figure 2 is~~ <u>Figures 2A and 2B show</u> a block diagram of the functional units, data paths and register files of Figure 1;

Rewrite the paragraph at page 8, line 16 to page 9, line 5 as follows:

A4

In microprocessor 1 there are shown a central processing unit (CPU) 10, data memory 22, program memory 23, peripherals 60 and an external memory interface (EMIF) with a direct memory access (DMA) 61. CPU 10 further has an instruction fetch/decode unit 10a-c , a plurality of execution units, including an arithmetic and load/store unit D1, a multiplier M1, an ALU/shifter unit S1, an arithmetic logic unit ("ALU") L1, a shared multi-port register file 20a from which data are read and to which data are written. Decoded instructions are provided from the instruction fetch/decode unit 10a-c to the functional units D1, M1, S1, and L1 over various sets of control lines which are not shown. Data are provided to/from the register file 20a from/to to load/store ~~units~~ <u>unit</u> D1 over a first set of busses 32a, to multiplier M1 over a second set of

A4

busses 34a, to ALU/shifter unit S1 over a third set of busses 36a and to ALU L1 over a fourth set of busses 38a. Data are provided to/from the memory 22 from/to the load/store ~~units~~ unit D1 via a fifth set of busses 40a. Note that the entire data path described above is duplicated with register file 20b and execution units D2, M2, S2, and L2. Load/store unit D2 similarly interfaces with memory 22 via a set of busses 40b. Instructions are fetched by fetch unit 10a from instruction memory 23 over a set of busses 41. Emulation circuitry 50 provides access to the internal operation of integrated circuit 1 which can be controlled by an external test/development system (XDS) 51.

Rewrite the paragraph at page 9, lines 19 to 28 as follows:

A5

When microprocessor 1 is incorporated in a data processing system, additional memory or peripherals may be connected to microprocessor 1, as illustrated in Figure 1. For example, Random Access Memory (RAM) 70, a Read Only Memory (ROM) 71 and a Disk 72 are shown connected via an external bus 73. Bus 73 is connected to the External Memory Interface (EMIF) which is part of functional block 61 within microprocessor ~~42~~ 1. A Direct Memory Access (DMA) controller is also included within block 61. The DMA controller part of functional block 61 connects to data memory 22 via bus 43 and is generally used to move data between memory and peripherals within microprocessor 1 and memory and peripherals which are external to microprocessor 1.

Rewrite the paragraph at page 10, lines 1 to 7 as follows:

A6

A detailed description of various architectural features of the microprocessor of Figure 1 is provided in coassigned U.S. Patent ~~application S.N. 09/012,813 (TI-25311)~~ 6,182,203 and is incorporated herein by reference. A detailed description of enhancements and extensions included in the architecture of microprocessor 1 is provided in coassigned U.S. ~~Provisional~~ Patent application S.N. ~~60/183,527 (TI-30302, *Microprocessor with Improved Instruction Set Architecture*)~~ 09/702,096 and is incorporated herein by reference.

Rewrite the paragraph at page 10, lines 8 to 16 as follows:

A7

~~Figure 2 is~~ Figures 2A and 2B show a block diagram of the execution units and register files of the microprocessor of Figure 1 and shows a more detailed view of the buses connecting the various functional blocks. In this figure, all data busses are 32 bits wide, unless otherwise noted. There are two general-purpose register files (A and B) in the processor's data paths. Each of these files contains 32 32-bit registers (A0–A31 for register file A 20a and B0–B31 for register file B 20b). The general-purpose registers can be used for data, data address pointers, or condition registers. Any number of reads of a given register can be performed in a given cycle.

Rewrite the paragraph at page 12, lines 1 to 11 as follows:

A8

Most data lines in the CPU support 32-bit operands, and some support long (40-bit) and double word (64-bit) operands. Each functional unit has its own 32-bit write port into a general-purpose register file 20a or 20b (Refer to Figure 2A). All units ending in 1 (for example, .L1) write to register file A 20a and all units ending in 2 write to register file B 20b. Each functional unit has two 32-bit read ports for source operands *src1* and *src2*. Four units (.L1, .L2, .S1, and .S2) have an extra 8-bit-wide port for 40-bit long writes, as well as an 8-bit input for 40-bit long reads. Because each unit has its own 32-bit write port, when performing 32-bit operations all eight units can be used in parallel every cycle. Since each multiplier can return up to a 64-bit result, two write ports (dst1 and dst2) are provided from the multipliers to the respective register file.

Rewrite the paragraph at page 12, lines 12 to 20 as follows:

A9

Each functional unit reads directly from and writes directly to the register file within its own data path. That is, the .L1 unit 18a, .S1 unit 16a, .D1 unit 12a, and .M1 ~~units~~ unit 14a write to register file A 20a and the .L2 unit 18b, .S2 unit 16b, .D2 unit 12b, and .M2 ~~units~~ unit 14b write to register file B 20b. The register files are connected to the opposite-side register file's functional units via the 1X and 2X cross paths. These cross paths allow functional units from

A9 one data path to access a 32-bit operand from the opposite side's register file. The 1X cross path allows data path A's functional units to read their source from register file B. Similarly, the 2X cross path allows data path B's functional units to read their source from register file A.

Insert the following <u>new</u> paragraph at page 12, after line 20:

A10 All eight of the functional units have access to the opposite side's register file via a cross path. The .M1, .M2, .S1, and .S2 units' *src2* inputs are selectable between the cross path and the same side register file by appropriate selection of multiplexers 213-214, for example. On units .L1 and .L2, depending on the particular opcode, either the src1 or the src2 input, but not both simultaneously, is selectable between the cross path and the same-side register file by appropriate selection of multiplexers 211-212, for example. Cross path 1X bus 210 couples one input of multiplexer 211 for src1 input of .L1 unit 18a, multiplexer 212 for src2 input of .L1 unit 18a, multiplexer 213 for src2 input of .S1 unit 16a and multiplexer 214 for scr2 input of .M1 unit 14a. Multiplexers 211, 212, 213, and 214 select between the cross path 1X bus 210 and an output of register file A 20a. Buffer 250 buffers cross path 2X output to similar multiplexers for .L2, .S2, .M2, and .D2 units.

Insert the following <u>new</u> paragraph after the new paragraph above at page 12 that begins at line 20:

A11 S2 unit 16b may write to control register file 102 from its dst output via bus 220. S2 unit 16b may read from control register file 102 to its src2 input via bus 221.

Rewrite the paragraph at page 13, lines 8 to 17 as follows:

A12 Bus 40a has an address bus DA1 which is driven by mux 200a. This allows an address generated by either load/store unit D1 or D2 to provide a memory address for loads or stores for register file 20a. Data Bus LD1 loads data from an address in memory 22 specified by address bus DA1 to a register in load unit D1. Unit D1 may manipulate the data provided prior to storing

A12

it in register file 20a. Likewise, data bus ST1 stores data from register file 20a to memory 22. Load/store unit D1 performs the following operations: 32-bit add, subtract, linear and circular address calculations. Load/store unit D2 operates similarly to unit D1 via bus 40b, with the assistance of mux 200b for selecting an address.

Rewrite the paragraph at page 14, lines 2 to 10 as follows:

A13

A set of basic instructions included in DSP 10 is described in U.S. Patent 6,182,203, ~~application S.N. 09/012,813 (TI-25311,~~ incorporated herein by reference~~)~~. A set of extended instructions is included in this embodiment of the present invention, as described in U.S. ~~Provisional~~ Patent application 09/702,096, ~~S.N. 60/183,527 (TI-30302,~~ incorporated herein by reference~~)~~. Alternative embodiments of the present invention may have different sets of instructions and functional unit mapping. The instruction sets discussed above are illustrative and are not exhaustive or intended to limit various embodiments of the present invention.

Rewrite the paragraph at page 20, lines 8 to 16 as follows:

A14

Figure 5 is a flow chart illustrating a subroutine call in a prior art DSP, a TMS320C62xx produced by Texas Instruments Incorporated. During execution of a program sequence 500, a subroutine 510 is called by executing a branch instruction 501 that provides ~~a~~ an address displacement "func" to subroutine 510 relative to the current contents of a program counter that is used to provide the addresses of program sequences. As is known, address displacement "func" is added to the program counter and program execution branches to the address associated with the address displacement value, as indicated by arc 512.

Rewrite the paragraph at page 20, lines 17 to 26 as follows:

A15

In this prior art DSP, there is no CALL instruction that automatically forms a return address. Therefore, a return address is formed by executing a move constant instruction (MVK) 502 that moves the least significant half of the address "LABEL" into the least significant half of

A15

general purpose register B3, and by executing a move constant instruction high (MVKH) ~~502~~ 503 that moves the most significant half of the address "LABEL" into the most significant half of general purpose register B3. At the completion of subroutine 510, an indirect branch instruction is executed that branches to the address indicated by a source register, in this case register B3, as indicated by arc 514.

Rewrite the paragraph at page 21, lines 4 to 13 as follows:

A16

Figure 6A is a flow chart illustrating a subroutine call with a return address computed using an "add constant to program counter" (ADDKPC) instruction 602, according to an aspect of the present invention. During execution of a program sequence 600, a subroutine 610 is called by executing a branch instruction 601 that provides ~~a~~ an address displacement "func" to subroutine 610 relative to the current contents of a program counter that is used to provide the addresses of program sequences. As is known, address displacement "func" is added to the program counter and program execution branches to the address associated with the address displacement value, as indicated by arc 612.

Rewrite the paragraph at page 22, lines 3 to 17 as follows:

A17

Another aspect of the ADDKPC instruction is that a parameter field 602c is provided that specifies a number of delay slots that are to be no-op'ed with virtual NOP instructions. In this embodiment of the present invention, a branch instruction has five delay slots, as discussed above. Therefore, up to five instructions will be executed prior to execution of the target instruction of a branch instruction. If no meaningful instructions can be placed in the delay slots, then NOP instructions must be placed there, or a multi-cycle-NOP instructions, such as instruction 504. In this example, parameter field 602c contains the number "4" such that four delay slots will be traversed without dispatching an instruction since the ADDKPC instruction 602 occupies one delay slot. In a similar manner, a branch with NOP instruction (BNOP) provides a parameter field that specifies a number of delay slots that are to be no-op'ed with virtual NOP instructions, as described in more detail in US Patent Application S.N. ________

A17 ~~(TI-26738) entitled *VLIW With Code Size Reduction Instructions: BNOP*~~ 09/702,484 and is incorporated herein by reference.

Rewrite table 7 at page 23, lines 1 to 3 as follows:

A18

Table 7 – ADDKPC Instruction Format

| format | ADDKPC (~~.u nit~~ .unit), src1, ~~src2,~~ dst, src2<br>Where:<br>Src1 is a seven bit signed constant<br>Src2 is ncnt, a 3 bit unsigned constant that defines a parallel number of NOPs<br>Dst is a register | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| field | 31-29 | 28 | 27-23 | 22-16 | 15-13 | 12 | 11-2 | 1 | 0 |
| Name | Creg | Z | Dst | disp | ncnt | 0 | Opcode | s | p |
| Width | 3 | 1 | 5 | 7 | 3 | 1 | 10 | 1 | 1 |

Rewrite the paragraph at page 23, lines 4 to 19 as follows:

A19 The displacement field (disp, a 7-bit constant) is shifted left by two and then added to the address ~~of the~~ of the fetch packet containing the ADDKPC instruction AND'ed with ~~-31~~ INV(31) (invert bits of binary value of 31). The result is placed in the destination register. In this embodiment a delayed version of the fetch packet program counter (PCF) is maintained that corresponds to the address of the instruction that is currently in the first execute phase of the pipeline (E1) and is referred to as PCE1. PCE1 represents the address of the first instruction of the fetch packet containing the instruction being executed. It is the same as the address of this ADDKPC instruction truncated to a fetch packet boundary (32 bytes). Note that if the execute packet spans two fetch packets, the instructions in the second fetch packet will see the value of the PCE1 register as 32 greater than the value seen by the first instruction in the execute packet. The operation of PCE1 will be described in more detail with reference to Figures 7-9. An assembler/linker can automatically compute the correct value for disp by the following formula:

disp = (address(LABEL) - (address(this_instruction) & ~~-31~~ INV(31))) >> 2.

Rewrite the paragraph at page 25, lines 1 to 15 as follows:

A20

Figure 7 is an illustration of execution packets spanning fetch packets for the processor of Figure 1. Advantageously, in the present embodiment of processor 10, an execution packet can cross an eight–word fetch packet boundary, thereby eliminating a need to add NOP instructions to pad fetch packets. For example, eight-word execution packet EP1 completely occupies fetch packet 700. Four-word execution packet EP2 partially fills fetch packet ~~702~~ 701. Six-word execution packet EP3 does not fit completely within fetch packet ~~702~~ 701, however, the first four words EP3(0)-EP3(3) are placed in fetch packet ~~702~~ 701 and the last two words EP3(4), EP3(5) are placed in fetch packet ~~704~~ 702. Therefore, the last *p* bit in a fetch packet is not always set to 0 in processor 10. If the last p-bit of a fetch packet is not zero, then instruction fetch control circuitry in stage 10a (Figure 1) fetches a second fetch packet and extracts instruction words until a p bit set to 0 is encountered. This sequence of instruction words is then ordered into a single execution packet, such as execution packet EP3, for example.

Rewrite the paragraph at page 25, line 28 to page 26, line 12 as follows:

A21

Execution packet 823 with instruction words 823(0)-823(1) is located within fetch packet 811 after instruction word 822(0). Note that the last word of the fetch packet does not necessarily have the p-bit set to 0, for example instruction words 821(0) and 824(0) both have their p-bit set to one indicating that their associated execution packet spans to the next fetch packet. Thus, instruction words 824(1-3) are located in fetch packet 812. Likewise, execution packet 826 spans fetch packets 812 and 813 with instruction words 826(0-2) located in fetch packet 812 after instruction words 825(0-1) and instruction words 826(3-7) located in fetch packet 813. In this example, instruction word 826(6) is an ADDKPC instruction. However, since instruction 826(6) is in the second fetch packet 813 of execution 826, PCE1 will contain an address associated with fetch packet 813 when instruction 826(6) reaches the first execute phase of unit S2. U.S. Patent Application ~~No.~~ S.N. ~~_______ (TI-30559) entitled *Microprocessor With Execution Packet Spanning Two Or More Fetch Packets*~~ 09/702,320 provides further details and is incorporated herein by reference.

Rewrite the paragraph at page 29, lines 12 to 15 as follows:

---

A22

Three multi-channel buffered serial ports (McBSP) 1060, 1062, 1064 are connected to DMA controller 1040. A detailed description of a McBSP is provided in U.S. Patent ~~S.N. 09/055,011 (TI-26204, Seshan, et al)~~ 6,167,466 and is incorporated herein by reference.

---

Rewrite the paragraph at page 30, lines 19 to 26 as follows:

---

A23

Figure 11 illustrates an exemplary implementation of an example of an integrated circuit ~~40~~ that includes digital system ~~1000~~ 40 in a mobile telecommunications device 15, such as a wireless telephone with integrated keyboard 12 and display 14. As shown in Figure 11 digital system ~~1000~~ 40 with processor ~~1001~~ (not shown) is connected to the keyboard 12, where appropriate via a keyboard adapter (not shown), to the display 14, where appropriate via a display adapter (not shown) and to radio frequency (RF) circuitry 16. The RF circuitry 16 is connected to an aerial 18.

---

Rewrite the **Abstract** at page 39, lines 4 to 21 as follows:

---

A24

A data processing system with a microprocessor ~~(10). The microprocessor~~ that has ~~in~~ an instruction execution pipeline that includes fetch and decode stages and several functional execution units ~~(L1/2, S1/2, M1/2, D1/2)~~. Fetch packets contain a plurality of instruction words. Execution packets include a plurality of instruction words that can be executed in parallel by two or more execution units. An execution packet can span two or more fetch packets. An add (k)constant to program counter (ADDKPC) instruction is provided, such that a parameter specified by the ADDKPC instruction is combined with a value provided by a program counter of microprocessor. The ADDKPC instruction can also specify a number of delay slots after a branch instruction to be filled with virtual NOP instructions such that memory is not wasted with useless NOP instructions. An ADDKPC instruction can provide a relative address for use as a return address. A plurality of predicated ADDKPC ~~instruction~~ instructions can provide a return

A24

address selected from a plurality of return ~~address~~ addresses. A compiler can reorder code with an ADDKPC instruction to absorb useless NOP instructions.

~~Figure 9~~